**WE CLAIM:**

1. A method of reducing unnecessary barrier instructions in a compiler for generating a parallel processing object program from a source program, comprising the steps of:

transforming with said compiler said source program into parallel processing execution divisions

issuing a post dynamic barrier instruction immediately before a reference position following said parallel processing execution divisions, said post dynamic barrier instruction having parameters with information used for determining the necessity of a barrier for a variable reference or array reference to be referenced after said parallel processing execution division,

determining whether there is data dependency present for a variable or array in said parallel processing execution division and said variable reference or array reference to be referenced after said parallel processing execution division based on the parameters of said post dynamic barrier instruction, and

dynamically reducing unnecessary barrier instructions in accordance with said determining whether data dependency is present.

2. A method of reducing unnecessary barrier instructions according to claim 1, further including issuing, immediately before said parallel processing execution divisions, a pre

dynamic barrier instruction having parameters for determining whether a barrier instruction is issued by said compiler, wherein said determining of whether there is data dependency present for a variable or array in said parallel processing execution division and said variable reference or array reference to be referenced after said parallel processing execution division is based on the parameters of said pre dynamic and said post dynamic barrier instructions.

3. A method of reducing unnecessary barrier instructions according to claim 2, further including, following said step of determining whether there is data dependency, checking if a memory access instruction in the parallel processing execution division has been issued and issuing a barrier instruction if said memory access instruction has not been issued.

4. A method of reducing unnecessary barrier instructions according to claim 2, further including, following said step of determining whether there is data dependency, checking if a memory access instruction for the data dependency has been issued and determining whether said memory access instruction has been completed, wherein if said memory access instruction has been issued and has been completed, a barrier instruction is not issued by said compiler.

5. A method of reducing unnecessary barrier instructions in a multiprocessor system including a compiler for generating a parallel processing object program from a source program, including converting said source program into parallel

processing objects having parallel processing loops, issuing a predynamic barrier instruction immediately before at least one of said parallel processing loops that has at least one parameter designating information about a first variable or array reference in said one loop, inserting a reference position before at least one second variable or array reference following said parallel processing loop and issuing a post dynamic barrier instruction immediately before said reference position having at lest one parameter with information about said second variable or array reference and

determining whether there is a data dependency present for said first variable or array based on the parameters of said pre dynamic and post dynamic barrier instructions, and

dynamically reducing unnecessary barrier instructions in accordance with said determining whether data dependency is present.

6. A method of reducing unnecessary barrier instructions according to claim 5, wherein in said step of dynamically reducing unnecessary barrier instructions, a barrier instruction is not issued by said compiler if no data dependency is determined to be present for said first variable or array reference according to said determining step.

7. A method of reducing unnecessary barrier instructions according to claim 5, wherein in said step of dynamically reducing unnecessary barrier instructions, determining if a memory access instruction for the data dependency is issued

and if said memory access instruction for data dependency is determined to be issued, further determining if said memory access instruction has been completed, wherein a barrier instruction is not issued by said compiler if said memory access instruction has been completed.

8. A method for reducing unnecessary barrier instructions according to claim 7, further including specifying, for each of said first and second variable or array reference, respectively, a base address, a stride width and an element length as said parameters for said pre dynamic post dynamic barrier instructions, respectively.

9. A multiprocessor system including a compiler for generating a parallel processing object program from a source program, said compiler converting said source program into parallel processing objects having parallel processing loops, issuing a pre dynamic barrier instruction immediately before at least one of said parallel processing loops that has at least one parameter designating information about a first variable or array reference in said one loop, inserting a reference position before a second variable or array reference following said parallel processing loop and issuing a post dynamic barrier instruction immediately before said reference position having at lest one parameter with information about said second variable or array reference; and

a dynamic barrier executing device that determines whether there is a data dependency present for said first

variable or array based on the parameters of said pre dynamic and post dynamic barrier instructions,

wherein said compiler dynamically reduces unnecessary barrier instructions in accordance with said determination of whether data dependency is present made by said dynamic barrier executing device.

10. A multiprocessor system according to claim 9, wherein said compiler does not issue a barrier instruction if said dynamic barrier executing device determines that no data dependency is present for said first variable or array reference.

11. A multiprocessor system according to claim 9, wherein said dynamic barrier executing device determines if a memory access instruction for the data dependency is issued and if said memory access instruction for data dependency is determined to be issued, further determines if said memory access instruction has been completed, wherein said compiler does not issue a barrier instruction if said memory access instruction has been completed.

12. A method of reducing unnecessary barrier instructions in a multiprocessor system including a compiler for generating a parallel processing object program from a source program, including:

converting said source program into parallel processing objects having parallel processing loops,

issuing a pre dynamic barrier instruction immediately before at least one of said parallel processing loops that has at least one parameter designating information about a first variable or array reference in said one loop,

inserting a reference position before a second variable or array reference following said parallel processing loop,

issuing a post dynamic barrier instruction immediately before said reference position having at least one parameter with information about said second variable or array reference

determining whether there is a data dependency present for said first variable or array based on the parameters of said pre dynamic and post dynamic barrier instructions, and

inserting a conditional branch statement and a barrier instruction in said parallel processing object after said one loop, said conditional branch instruction having an argument that is satisfied if the data dependency is determined to be not present, wherein said branch is followed to prevent said barrier instruction from being executed when said argument of said conditional branch statement is satisfied.

13. A computer readable storage medium encoded with executable instructions representing a compile program that generates a parallel processing object program from a source program, said compile program comprising:

converting said source program into parallel processing objects having parallel processing loops, issuing a pre dynamic barrier instruction immediately before at least one of

said parallel processing loops that has at least one parameter designating information about a first variable or array reference in said one loop, inserting immediately before a reference position for at least one second variable or array reference following said parallel processing loop a post dynamic barrier instruction having at lest one parameter with information about said second variable or array reference,

determining whether there is a data dependency present for said first variable or array based on the parameters of said pre dynamic and post dynamic barrier instructions, and

dynamically reducing unnecessary barrier instructions in accordance with said determining whether data dependency is present.

14. A computer readable storage medium encoded with executable instructions representing a compile program according to claim 13, wherein a barrier instruction is not issued by said compiler program if no data dependency is determined to be present for said first variable or array reference according to said determining step.

15. A computer readable storage medium encoded with executable instructions representing a compile program according to claim 13, including determining if a memory access instruction for the data dependency is issued and if said memory access instruction for data dependency is determined to be issued and further determining if said memory access instruction has been completed, wherein a barrier

instruction is not issued by said compiler program if said memory access instruction has been completed.

16. A computer readable storage medium encoded with executable instructions representing a compile program according to claim 15, further including specifying, for each of said first and second variable or array reference, respectively, a base address, a stride width and an element length as said parameters for said pre dynamic post dynamic barrier instructions, respectively.

17. A computer readable storage medium encoded with executable instructions representing a compile program that generates a parallel processing object program from a source program, said compile program comprising:

converting said source program into parallel processing objects having parallel processing loops,

issuing a pre dynamic barrier instruction immediately before at least one of said parallel processing loops that has at least one parameter designating information about a first variable or array reference in said one loop,

inserting a reference position before a second variable or array reference following said parallel processing loop,

issuing a post dynamic barrier instruction immediately before said reference position having at least one parameter with information about said second variable or array reference

determining whether there is a data dependency present for said first variable or array based on the parameters of

said pre dynamic and post dynamic barrier instructions, and

inserting a conditional branch statement and a barrier instruction in said parallel processing object after said one loop, said conditional branch instruction having an argument that is satisfied if the data dependency is determined to be not present, wherein said branch is followed to prevent said barrier instruction from being executed when said argument of said conditional branch statement is satisfied.